UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,834	10/24/2003	Trishul Chilimbi	3382-66135-01	1003

26119 7590 11/12/2008
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET
SUITE 1600
PORTLAND, OR 97204

EXAMINER	
WEI, ZHENG	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
11/12/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/693,834	**Applicant(s)** CHILIMBI ET AL.	
	Examiner ZHENG WEI	**Art Unit** 2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *21 July 2008*.

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-10 and 12-25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-10 and 12-25* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) □ Notice of References Cited (PTO-892)

2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date *07/21/2008*.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) □ Notice of Informal Patent Application

6) □ Other: _____.

DETAILED ACTION

Remarks

1. This office action is in response to the amendment filed on 07/21/2008.
2. Claim 11 has been canceled.
3. Claims 1, 6, 7, 18 and 23 have been amended.
4. The 35 U.S.C. 112 second paragraph rejection to claims 6, 7-10, 12, 13 and 18-25 is withdrawn in view of the Applicant's amendment.
5. Claims 1-10 and 12-25 remain pending and have been examined.

Information Disclosure Statement

6. The information disclosure statements filed on 07/21/2008 has been placed in the application file and the information referred to therein has already been considered.

Summary of Communications

7. The Examiner contacted the Attorney Stephen A. Wight (Reg#:37, 759) on 10/31/2008 to discuss the case and proposed an amendment on 11/03/2008. The Attorney Wight and Ryan Fox proposed another version of amendment during a phone interview on 11/06/2008. The Examiner pointed out that the Applicants' proposal addressed the Examiner's suggestions but further included new independent claims which had not been claimed and considered before. Therefore, it did not put the claimed application as a whole in the allowable position. The Examiner contacted Attorney Wight again on 11/07/2008 tried to reach an

agreement based on latest proposed amendment made on 11/07/2008. During the phone conversation, Mr. Wight further provided supported specification for the new claims and wanted keep the claims as they proposed. Thus, the agreement has not been reached based on the Examiner's proposal because the new claims had not been claimed and considered before and there are no drawings in the application to support the newly added claims. Therefore, all of the versions of the proposed amendments will be put in record file but will not be entered. This office action is based on the Applicant's response filed on 07/21/2008.

Response to Arguments

8. Applicant's arguments filed on 07/21/2008, in particular on pages 8-18, have been fully considered.

- The double patenting rejection to claim 1, 6, 18 and 23 in previous office is withdrawn in view of the Applicants argument, but a new ground of double patenting rejection is applied over claim 1 of U.S. Patent No. 7,140,008 in view of **Chilimbi** (Chilimbi et al., "Dynamic Hot Data Stream Prefetching for General-Purpose Programs", 2002).
- Another double patenting rejection to Claims 1, 6, 13, 18 and 23 is applied based on the latest amendment filed for a copending Application No. 10/892,260. Therefore, claims 1, 6, 13, 18 and 23 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over

claims of copending Application No. 10/892,260 in view of **Chilimbi** (Chilimbi et al., "Dynamic Hot Data Stream Prefetching for General-Purpose Programs", 2002).

- The rejection to claims 1-10 and 12-25 is withdrawn in view of the Applicants arguments. A new ground rejection is applied in further view of **Chilimbi** (Chilimbi et al., "Dynamic Hot Data Stream Prefetching for General-Purpose Programs", 2002).

Double Patenting

9. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement. Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

10. Claims 1, 6, 18 and 23 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claim 1 of U.S. Patent No. 7,140,008 in view of **Chilimbi** (Chilimbi et al., “Dynamic Hot Data Stream Prefetching for General-Purpose Programs”, 2002) as listed as an example in the table below. Although the conflicting claims are not identical, they are not patentably distinct from each other because limitations in one claim can obviously be applicable in the corresponding claim in view of **Chilimbi**.

Claims 1, 6, 18 and 23:

Claims 1, 6, 18 and 23 of the instant application contains every element of Claim 1 of the US patent as shown in the table below. The only difference is that the switch of the execution between instrumented version and original version code path or procedure based a sampling rate while the claim 1 of the patent switching according to the count parameter.

However, **Chilimbi** in the same analogous art of busty tracing discloses using sampling rate to reduce the overhead of profiling (see for example Figure 2; p. 201, left column, "A common way to reduce the overhead of profiling is sampling: instead of recording all data references, sample a small, but representative fraction of term. Our profiler obtains a temporal profile with low overhead by sampling bursts of data reference, which are subsequences of the reference trace "). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the sampling rate in **Chilimbi** with claim 1 of the patent. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce the overhead of profiling as suggested by **Chilimbi**.

11. Although the conflicting claims are not identical, they are not patentably distinct from each other. As can be seen from the table below, instant claims and the claims of U.S. Patent are directed to the same subject matter of the invention. For example,

Instant Application 10/693,834	**U.S. Patent** 7,140,008
Claim 1. **A method of instrumenting a program** to provide instrumentation data, the method comprising:	Claim 1. **A method of instrumenting a program** to provide sampled temporal profiling bursts of a program execution trace, the method comprising:
creating an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data;	**providing a duplicate version of at least some procedures in the program** with instrumentation for capturing a temporal sequence of data references by the program;

	inserting check code at locations of at least some procedure entries and loop back-edges of the program;
tracking a frequency of execution of the code paths;	alternately **tracking a number of iterations** of the check code executed in a checking phase and a profiling phase up to respective checking and profiling count parameters, wherein the profiling count parameter is more than one and the duplicate version of at least some procedures with instrumentation are executed during the profiling phase and a non-instrumented version of the program's procedures are executed during the checking phase;
when a code path is to be executed, **determining to** dispatch execution into the instrumented version code path at a sampling rate for the respective code path and otherwise into the original version code path such that, for a given sampling rate, a ratio of a number of executions of the instrumented version code path to a total number of executions of both the instrumented version code path and the original version code path is equivalent to the given sampling rate.	upon executing the check code when in **the checking phase**, causing execution to proceed in the non-instrumented version of the program's procedures; upon executing the check code when in the profiling phase, causing execution to proceed in the duplicate instrumented version of the at least some procedures; and
adapting the sampling rate for the code paths according to the frequency of execution of the code paths, such that, after adapting, a ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the adjusted sampling rate	**switching between checking and profiling phases upon the tracked number of iterations of the check code reaching the respective count parameter of the respective phase.**

Claim 6.	Claim 1.
A method of instrumenting a computer program containing procedures, the method comprising:	**A method of instrumenting a program** to provide sampled temporal profiling bursts of a program execution trace, the method comprising:
creating a copy of at least some of the original procedures in the computer program;	**providing a duplicate version of at least some procedures in the program** with instrumentation for capturing a temporal sequence of data references by the program;
inserting instrumentation into the copies; creating an executable version of the program containing the original procedures and the copies;	**inserting check code** at locations of at least some procedure entries and loop back-edges of the program;
	alternately tracking a number of iterations of the check code executed in a checking phase and a profiling phase up to respective checking and profiling count parameters, wherein the profiling count parameter is more than one and the duplicate version of at least some procedures with instrumentation are executed during the profiling phase and a non-instrumented version of the program's procedures are executed during the checking phase;
executing the executable version of the program, wherein the copies of the procedures are executed in bursts, and the frequency at which the bursts are performed decreases as the total number of executions of **either the original**	upon **executing the check code** when in the checking phase, causing execution to proceed in the **non-instrumented version** of the program's procedures; upon executing the check code when in the profiling phase, causing execution to

procedure or copy of the procedure is executed, said frequency comprising the number of executions of the copy of the procedure divided by the total number of executions.	proceed in **the duplicate instrumented version** of the at least some procedures; and
	switching between checking and profiling phases upon the tracked number of iterations of the check code reaching the respective count parameter of the respective phase.
Claims 18, 23. **A method of instrumenting software**, the method comprising:	Claim 1. **A method of instrumenting a program** to provide sampled temporal profiling bursts of a program execution trace, the method comprising:
producing a copy of at least some procedures of the software;	**providing a duplicate version of at least some procedures in the program** with instrumentation for capturing a temporal sequence of data references by the program;
inserting instrumentation into the copies; and	**inserting check code** at locations of at least some procedure entries and loop back-edges of the program;
sampling a copy of a procedure at a rate inversely proportional to how frequently either the original or the copy of the procedure is executed/**sampling a copy of a procedure** at higher rates for procedures executed less frequently and sampling a copy of a procedure at lower rates for procedures executed more frequently.; wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure.	alternately **tracking a number of iterations of the check code executed in a checking phas**e and a profiling phase up to respective checking and profiling count parameters, wherein the profiling count parameter is more than one and the duplicate version of at least some procedures with instrumentation are executed during the profiling phase and a non-instrumented version of the program's procedures are executed during the checking phase;
	upon executing the check code when in the checking phase, causing execution to

	proceed in the non-instrumented version of the program's procedures; upon executing the check code when in the profiling phase, causing execution to proceed in the duplicate instrumented version of the at least some procedures; and switching between checking and profiling phases upon the tracked number of iterations of the check code reaching the respective count parameter of the respective phase.

12. Claims 1, 6, 13,18 and 23 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims of copending Application No. 10/892,260 in view of **Chilimbi** (Chilimbi et al., “Dynamic Hot Data Stream Prefetching for General-Purpose Programs”, 2002) as **listed as an example** in the table below. Although the conflicting claims are not identical, they are not patentably distinct from each other because limitations in one claim can obviously be applicable in the corresponding claim in view of **Chilimbi**.

Claims 1, 6, 13, 18 and 23:

Claims 1, 6, 13, 18 and 23 of the instant application contains every element of Claims 7, 15 and 22 of the co-pending application as shown in the table below. The only difference for claims 7 and 15 is that the instant claims further provide a definition of sampling rate while the co-pending claims only includes a sampling rate. (For the co-pending application, it recites a count parameter which is the same as being rejected as addressed above). However, **Chilimbi** in the same analogous art of busty tracing discloses using sampling rate to reduce the overhead of profiling (see for example Figure 2; p. 201, left column, "A common way to reduce the overhead of profiling is sampling: instead of recording all data references, sample a small, but representative fraction of term. Our profiler obtains a temporal profile with low overhead by sampling bursts of data reference, which are subsequences of the reference trace "). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the sampling rate in **Chilimbi** with claim 1 of the patent. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce the overhead of profiling as suggested by **Chilimbi**.

13. This is a provisional obviousness-type double patenting rejection. Since the conflicting claims have not in fact been patented. For example

Instant Application 10/693,834	**Co-pending Application** 10/892,260
Claim 1.	Claim 7:
A method of instrumenting a program to provide instrumentation data, the method comprising:	**A method of testing** for memory leaks during an execution of a program, the method comprising: Providing the program to be executed;
creating an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data;	**Providing a duplicate version of at least some procedures in the program** with instrumentation for capturing a temporal sequence of data references by the program the duplicate version of at least some procedures in the program being duplicate instrumented code;

tracking a frequency of execution of the code paths;	Inserting check code at locations of at least some dispatch placement points; Causing execution to proceed in the program upon executing the check code when in a checking phase; **Switching from the checking phase to the profiling phase at sampling rate** adapted in inverse proportion to an **execution frequency of a respective code path**; Tracking a staleness of an object of the program; and
when a code path is to be executed, **determining to** dispatch execution into the instrumented version code path at a sampling rate for the respective code path and otherwise into the original version code path such that, for a given sampling rate, a ratio of a number of executions of the instrumented version code path to a total number of executions of both the instrumented version code path and the original version code path is equivalent to the given sampling rate.	Running the program.
adapting the sampling rate for the code paths according to the frequency of execution of the code paths, such that, after adapting, a ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the adjusted sampling rate	

Claim 7:	Claim 15:
A method for detecting memory leaks in software, the method comprising:	A low-overhead method of detecting memory leaks, the method comprising:
Creating an instrumented version of the software containing an original version and **an instrumented copy version** of each procedure in the software;	Tracing an execution of a program by switch between a checking code and an instrumented code, the checking code being a **duplicate version** of the program modified to contain check code at least some dispatch check placement points, the instrumented code being a duplicate version of at least some procedures of the program, modified to contain instrumentation for capturing a temporal sequence of data references by the program;
Executing the instrumented version of the software, wherein the instrumented version of the procedures **are sampled** at higher rates for procedures whose total number of executions of both the original version and the copy version are executed **less frequently** and sampled at lower rates for procedures whose total number of executions of both the original versions and the copy versions are executed **more frequently**, wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure;	Causing execution to proceed in the checking code upon executing the check code when in a profiling phase; Causing execution to proceed in the instrumented code upon executing the check code when in a profiling phase;
storing instrumentation data obtained by execution of the instrumented version of the software; and	Switching from the checking phase to the profiling phase **at a sampling rate adapted in inverse proportion to an execution frequency of a respective code path; and**
Reporting all objects that satisfy a predefined staleness condition as **memory leaks**	Tracking staleness of an object of the program.

Claim Rejections - 35 USC § 112

14. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

15. Claims 12 and 18-22 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 12:

Claim 12 recites the limitation “the heap” in lines 2. There is insufficient antecedent basis for this limitation in the claim.

Claim 18:

Claim 18 recites the limitation “the original” in line 6. There is insufficient antecedent basis for this limitation in the claim.

Claims 19-22:

Claims 19-22 depend on claim 18. Therefore, they are also rejected for the same reason.

Claim Rejections - 35 USC § 103

16. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

17. Claims 1-6, 14, 18, 19 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wu (Youfeng Wu, US 7,032,217 B2) in view of Chilimbi (Chilimbi et al., “Dynamic Hot Data Stream Prefetching for General-Purpose Programs”, 2002)

Claim 1:

Wu discloses a method of instrumenting a program to provide instrumentation data, the method comprising:

- creating an instrumented version of the program comprising duplicate versions of at least some code paths in the programs and an instrumented version code path with instrumentation code for capturing instrumentation data (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- tracking a relative frequency of execution of the code paths (see for example, Fig.5A, Fig.5B, steps 505-530, steps 555-585 and related text about “profile Counter”);
- when a code path is to be executed, determining to dispatch execution into the instrumented version code path at a sampling rate for the respective code path and otherwise into the original version code path, (see for example, Fig.5A, Fig.5B, steps 530, 585 and related text; also see col.8, lines 36-38, “profiles(e)” “counter(e)”, where the “counter(e)” is the profile counter value collected by the profiling hardware and “profile(e)” is the edge frequency derived by the dynamic optimizer from counter(e)”) and

- adapting the sampling rate for the code paths according to the relative frequency of execution of the code paths, such that, after adapting, a ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the adjusted sampling rate. (see for example, Fig.5A, Fig.5B, steps 530, 585 and related text about "Trigger Counter", "counter(e)" and "profile(e)"; also see col.8, lines 62-63, "Once a profiling instruction is executed, the profile counters are updated at processing block 515").

But does not explicitly disclose an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data and the given sampling rate, the ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the given sampling rate. However, Chilimbi discloses such limitations (see for example, Figure 2, "checking code" and "instrumented code" and related text; also see "sampling rate" p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi's method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column).

Claim 2:

Wu further discloses the method of claim 1 wherein instrumentation data comprises data relating to runtime data references, branch executions, memory allocations, synchronization events, data loads, data stores, or branches (see for example, Fig.3, element 320 and related text, also see p.3, line1 “three branch instructions”).

Claim 3:

Wu discloses a method of instrumenting a program to provide runtime program data, the method comprising:

- providing a instrumented version of at least some already present procedures in the program with instrumentation for capturing runtime program data (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- executing the duplicate version of at least some of the procedures (see for example, Fig.2, step 220, “Program execution” and related text); and
- subsequently, for each of the already resent procedures, selectively reducing the frequency at which the duplicate version of the procedure is executed (see for example, Fig.5A, step 525 “Decrement Trigger Counter” and related text; also see col.8, lines 36-38, “profiles(e), where the “counter(e)” is the

profile counter value collected by the profiling hardware and “profile(e)” is the edge frequency derived by the dynamic optimizer from counter(e)”).

But does not explicitly disclose an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data and the given sampling rate, the ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the given sampling rate. However, Chilimbi discloses such limitations (see for example, Figure 2, “checking code” and “instrumented code” and related text; also see “sampling rate” p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi’s method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column). But Wu does not explicitly disclose the frequency equivalent to the number of executions of the duplicate version taken as percentage of the total number of executions of the procedure. However, it is well known in the computer art, the frequency can be represented by different ways, such as percentage, ratio…Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use percentage or ratio to represent frequently executing of profile procedure by using “profile(e)” divided by total number of counter(e). One would have been motivated to use ratio to represent sample rate, because the terms ratio, percentage are widely used forms/units for rate.

Claim 4:

Wu further discloses the method of claim 3 wherein the frequency at which the duplicate version is executed is reduced at a rate inversely proportional to how frequently a procedure of the software is executed (see for example, Fig.5B, steps 570, 575, 576 and 580 “Decrement/Increment Counter” and related text).

Claim 5:

Wu also discloses the method of claim 3 wherein the frequency at which the duplicate version is executed is reduced as a function of how frequently a procedure of the software is executed (see for example, Fig.2, step 240, 250 and related text, also see Fig.5A, steps 54, 545, “Generate New Phase Transition Information” and related text).

Claim 6:

Wu discloses a method of instrumenting a computer program containing procedures, the method comprising:

- creating a copy of at least some of the original procedures in the computer program (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- inserting instrumentation into the copies (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- creating an executable version of the program containing the original procedures and the copies (see for example, Fig.2, steps 210-220 and related text, also see col.4, line 64-col.5, line 20 about “compiler”);
- executing the executable version of the program, wherein the copies of the procedures are executed in bursts, and the frequency at which the bursts are performed decreases as the total number of executions of either the original procedure or copy of the procedure is executed, (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20, also see Fig.5A, step 525, “Decrement Trigger Counter” and related text).

But does not explicitly disclose an instrumented version of the program comprising duplicate versions of at least some code paths in the programs, such that a duplicate code path has an original version code path and an instrumented version code path with instrumentation code for capturing instrumentation data and the given sampling rate, the ratio of a number of executions of the instrumented version code path to a total number of executions of the code path is equivalent to the given sampling rate. However, Chilimbi discloses such limitations (see for example, Figure 2, “checking code” and “instrumented code” and related text; also see “sampling rate” p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi’s method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column). But Wu does not explicitly disclose said frequency comprising the number of executions of the copy of the procedure divided by the total number of executions. However, Wu also discloses the “profiles(e)”, “counter(e)”, where the “counter(e)” is the profile counter value collected by the profiling hardware and “profile(e)” is the edge frequency derived by the dynamic optimizer from counter(e)” (see for example, col.8, lines 36-38). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to calculate the frequency by using the number of executions of the copy of the procedure divided by the total number of executions (profiles(e)/counter(e)). One would have been motivated to calculate frequency in such a way, because the ratio is the widely used forms/units for rate, frequency.

Claim 14:

Wu discloses a method of analyzing software, the method comprising:

- creating an instrumented version of the software containing an instrumented version of at least some procedures in the software, wherein the instrumented versions comprise instrumentation points (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);

- inserting additional programming code at the instrumentation points that produce runtime information when executed (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20); and
- executing the instrumented version of the software, wherein the additional programming code is executed more frequently when located at instrumentation points that are less frequently executed, and the additional programming code is executed less frequently when located at instrumentation points for procedures that are more frequently executed (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20, also see Fig.5A, step 525, "Decrement Trigger Counter" and related text);

But does not explicitly disclose an instrumented version of the program comprising an original version of some procedures in the software. However, Chilimbi discloses such limitations (see for example, Figure 2, "checking code" and "instrumented code" and related text; also see "sampling rate" p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi's method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column).

But Wu does not explicitly disclose wherein frequency of execution of additional programming code for a given procedure comprises a number of executions of the additional programming code taken as a percentage of total executions of the procedure. However, Wu also discloses the “profiles(e)”, “counter(e)”, where the “counter(e)” is the profile counter value collected by the profiling hardware and “profile(e)” is the edge frequency derived by the dynamic optimizer from counter(e)” (see for example, col.8, lines 36-38). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to calculate the frequency by using a number of executions of the additional programming code taken as a percentage of total executions of the procedure (profiles(e)/counter(e)). One would have been motivated to calculate frequency in such a way, because the ratio is the widely used forms/units for rate, frequency.

Claim 18:

Wu discloses a method of instrumenting software, the method comprising:

- producing a copy of at least some procedures of the software (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- inserting instrumentation into the copies (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20); and

- sampling a copy of a procedure at a rate inversely proportional to how frequently either the original or the copy of the procedure is executed (see for example, Fig.5A, Fig.5B, steps 530, 585 and related text about "Trigger Counter");

But does not explicitly disclose an instrumented version of the program comprising an original of the procedure. However, Chilimbi discloses such limitations (see for example, Figure 2, "checking code" and "instrumented code" and related text; also see "sampling rate" p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi's method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column).
But does not explicitly disclose wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure. However, Wu also discloses the "profiles(e)", "counter(e)", where the "counter(e)" is the profile counter value collected by the profiling hardware and "profile(e)" is the edge frequency derived by the dynamic optimizer from counter(e)" (see for example, col.8, lines 36-38). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to calculate the rate by using a number of executions of the instrumented version

of the procedure taken as a percentage of total number of executions of either version of the procedure. (profiles(e)/counter(e)). One would have been motivated to calculate rate in such a way, because the ratio in percentage is the widely used forms/units for rate, frequency.

Claim 19:

Wu further disclose the method of claim 18, wherein the instrumentation stores data relating to the software when executed (see for example, col.5, lines 40-41, “store the counter back to memory”).

Claim 23:

Wu discloses a method of instrumenting software, the method comprising:

- producing a copy of at least some procedures of the software (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- inserting instrumentation into the copies (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20); and
- sampling a copy of a procedure at higher rates for procedures whose original versions or copies are executed less frequently and sampling a copy of a procedure at lower rates for procedures whose original versions or copies are executed more frequently (see for example, Fig.5A, Fig.5B, steps 530, 585 and related text about “Trigger Counter”);

But Wu does not explicitly disclose an instrumented version of the program comprising original version and sampling rate. However, Chilimbi discloses such limitations (see for example, Figure 2, “checking code” and “instrumented code” and related text; also see “sampling rate” p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi’s method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column). But Wu does not explicitly disclose wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure. However, Wu also discloses the “profiles(e)”, “counter(e)”, where the “counter(e)” is the profile counter value collected by the profiling hardware and “profile(e)” is the edge frequency derived by the dynamic optimizer from counter(e)” (see for example, col.8, lines 36-38). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to calculate the rate by using a number of executions of the instrumented version of the procedure taken as a percentage of total number of executions of either version of the procedure (profiles(e)/counter(e)). One would have been motivated to calculate rate in such a way, because the ratio in percentage is the widely used forms/units for rate, frequency.

18. Claims 7-10, 12, 13 and 15-17are rejected under 35 U.S.C. 103(a) as being unpatentable over Wu (Youfeng Wu, US 7,032,217 B2) in view of Chilimbi (Chilimbi et al., “Dynamic Hot Data Stream Prefetching for General-Purpose Programs”, 2002) and further in view of Alexander (Alexander et al., US 6,658,652 B1).

Claim 7:

Wu discloses a method for detecting memory leaks in software, the method comprising:

- creating an instrumented version of the software containing an original version and an instrumented version of each procedure in the software (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20);
- executing the instrumented version of the software, wherein the instrumented version of the procedures are sampled at higher rates for procedures whose original versions or copies are executed less frequently and sampled at lower rates for procedures whose original versions or copies are executed more frequently, wherein a rate for a given procedure comprises a number of executions of the instrumented version of the procedure taken as percentage of total number of executions of either version of the procedure; (see for example, Fig.2, step 210 and related text, also see col.4, line 64-col.5, line 20, also see Fig.5A, step 525, “Decrement Trigger Counter” and related text);

- storing instrumentation data obtained by execution of the instrumented version of the software (see for example, Fig.4, element 460, “Profile operations buffer” and related text);

But Wu does not explicitly disclose an instrumented version of the program comprising original version, copy version and sampling rate. However, Chilimbi discloses such limitations (see for example, Figure 2, “checking code” and “instrumented code” and related text; also see “sampling rate” p.201, left column, the second paragraph from the bottom). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate Chilimbi’s method to trace and collect instrumentation data. One would have been motivated to do so to accomplish extremely low overhead for bursty tracing as suggested by Chilimbi (see for example, p.200, first paragraph of left column)

But Wu does not disclose reporting all objects that satisfy a staleness predicate as memory leaks. However, Alexander in the same analogous art of program tracing using shadow heap memory leak detection and other heap analysis discloses (see for example, Fig.30, JVM HEAP 3002, objects 3062-3068 and related text). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use Wu’s method to further implement memory leaks detection function to test memory leaks. One would have been motivated to do so as to provide a method for accurate memory leak detection in an object-oriented environment during real-time trace processing as

Alexander suggested at col.3, lines 33-37.

Claim 8:

Wu, Chilimbi and Alexander disclose the method of claim 7, Alexander further discloses, wherein instrumentation data comprises heap allocation, heap free and heap access information (see for example, Fig.3B, element 372 “Heap” and related text, also see Fig.31, step 3114 “Profiler finds the proper slot in the shadow heap based on the relative position of the corresponding object in the heap” and related text).

Claim 9:

Wu, Chilimbi and Alexander disclose the method of claim 7, Alexander further discloses, wherein reporting all objects comprises reporting the heap object, responsible allocation, heap frees that deallocated objects created at that allocation site, and the last access to the leaked object (see for example, Fig.32, steps 3202-3216, “Object deallocation” and related text).

Claim 10:

Wu, Chilimbi and Alexander disclose the method of claim 9, Alexander further discloses the method of claim 9, generating report for heap objects including the information of the last access to a leaked object (see for example, Fig.34, Fig.35, example reports and related text).

Claim 11:

Wu, Chilimbi and Alexander disclose the method of claim 7, Alexander further discloses the method comprising creating mapping information from the software to facilitate "last access" information (see for example, Fig.28 about data structure to facilitate tracking additional information related to a routine using heap and related text".

Claim 12:

Wu, Chilimbi and Alexander disclose the method of claim 7, Alexander further discloses, wherein the staleness predicate comprises determining whether an object on the heap has not been accessed within a predetermined length of time (see for example, Fig.10c-10D and related text).

Claim 13:

Wu, Chilimbi and Alexander disclose the method of claim 7, Wu further discloses, wherein the instrumented version of the procedures are sampled at a rate inversely proportional to how frequently a procedure is executed (see for example, Fig.5B, steps 570, 575, 576 and 580 “Decrement/Increment Counter” and related text).

Claims 15 and 17:

Wu and Chilimbi disclose the method of claim 14, but does not discloses, wherein runtime information comprises data relating to memory leaks or invariance. However, Alexander in the same analogous art of program tracing using shadow heap memory leak detection and other heap analysis discloses (see for example, Fig.30, JVM HEAP 3002, objects 3062-3068 and related text). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use Wu's method to further implement memory leaks detection function to test memory leaks. One would have been motivated to do so as to provide a method for accurate memory leak detection in an object-oriented environment during real-time trace processing as Alexander suggested at col.3, lines 33-37.

Claim 16:

Wu and Chilimbi disclose the method of claim 14, but does not discloses, wherein runtime information comprises data relating to data races. However, Alexander in the same analogous art of tracing software program discloses runtime information comprises data relating to data races (see for example, col.16, lines 27-36, "routine B" and its status: interrupt or suspended or blocked… and related text). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to further data race information that can be used to prevent data race.

19. Claims 20-22 and 24-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Wu (Youfeng Wu, US 7,032,217 B2) in view of Chilimbi (Chilimbi et al., "Dynamic Hot Data Stream Prefetching for General-Purpose Programs", 2002) and further in view of Zorn (Zorn et al., "A Memory Allocation Profiler for C and Lisp Programs")

Claim 20:

Wu and Chilimbi disclose the method of claim 19, but do not disclose the method further comprising providing the stored data to a tool for analysis. However, Zorn in the same analogous art of error detecting discloses a tool "mprof" which is used to study the memory allocation behavior of programs. Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to pass data to "mprof" for the purpose of monitor. One would have been motivated to do so to monitor executing programs and display to user as suggested by Zorn (p.1, Introduction, "allows programmer to identify where and why memory is being allocated in a program.")

Claim 21:

Wu, Chilimbi and Zorn disclose the method of claim 20, Zorn further discloses wherein the tool detects memory leaks (see for example, p.3, section 2 Using mprof).

Claim 22:

Wu, Chilimbi and Zorn disclose the method of claim 20, however, neither of them explicitly discloses the tool detects data races. However, it is well known in the computer art that the operating system which the tool is running provide the same functionality about data race detection and /or protection. Therefore, this claim is obvious by Wu, Chilimbi and Zorn.

Claim 24:

Wu and Chilimbi disclose the method of claim 23, but do not disclose the method further comprising providing the data to software to a tool. However, Zorn in the same analogous art of error detecting discloses a tool “mprof” which is used to study the memory allocation behavior of programs. Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to pass data to “mprof” for the purpose of monitor. One would have been motivated to do so to monitor executing programs and display to user as suggested by Zorn (p.1, Introduction, “allows programmer to identify where and why memory is being allocated in a program.”)

.Claim 25:

Wu, Chilimbi and Zorn disclose the method of claim 24, Zorn further discloses wherein the tool uses the communicated data to analyze the software (see for example, p.3, section 2 Using mprof).

Conclusion

20. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

21. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Zheng Wei whose telephone number is (571) 270-1059 and Fax number is (571) 270-2059. The examiner can normally be reached on Monday-Thursday 8:00-15:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner’s supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature of relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571- 272-1000.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Z. W./
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192